Group Round Robin: Improving the Fairness and Complexity of Packet Scheduling

Bogdan Caprita, Wong Chun Chan and Jason Nieh
Email: bc2008@columbia.edu, wc164@cs.columbia.edu, nieh@cs.columbia.edu
Department of Computer Science
Columbia University
Technical Report CUCS-018-03
June 2003

***Abstract*— We introduce Group Round-Robin (GRR) scheduling, a hybrid scheduling framework based on a novel grouping strategy that narrows down the traditional tradeoff between fairness and computational complexity. GRR combines its grouping strategy with a specialized round-robin scheduling algorithm that utilizes the properties of GRR groups to schedule flows within groups in a manner that provides $O(1)$ bounds on fairness with only $O(1)$ time complexity. Under the practical assumption that GRR employs a small constant number of groups, we apply GRR to popular fair queueing scheduling algorithms and show how GRR can be used to achieve constant bounds on fairness and time complexity for these algorithms.**

***Index Terms*— Stochastic Processes/Queueing Theory, Quality of Service, Scheduling, Fair Queueing**

I. Introduction

Packet scheduling is a important mechanism for meeting the quality-of-service requirements of competing flows in packet-switched data networks. An important class of packet schedulers are those that treat flows in a proportionally fair manner. Given a set of flows with associated weights, these schedulers try to allocate network link capacity to each flow in proportion to its respective weight. The goal of these schedulers is to provide the highest degree of fairness in allocating link capacity with the lowest time complexity of scheduler execution.

Many packet scheduling algorithms have been proposed with different trade-offs in fairness and time complexity. The fairest one is Generalized Processor Sharing (GPS) [9], which is an idealized fluid model that services flows continuously and simultaneously, but cannot be employed in practice since packets must be transmitted as a unit. Algorithms such as Weighted Fair Queueing (WFQ) [5] use a notion of virtual time to emulate GPS and approximate its behavior. However, WFQ can allow the service that a flow receives to deviate from service under GPS in the worst case by $O(N)$, where N is the number of flows being scheduled. Since GPS provides ideal max-min fairness, deviations in service from GPS are considered service error. Furthermore, WFQ requires at least $O(\log N)$ time complexity. Other variants of WFQ such as Virtual-clock [14], SFQ [8], SCFQ [6], SPFQ [12], and Time-shift FQ [4] have also been proposed. However, these algorithms share the same fairness and time complexity bounds as WFQ, allowing service errors of $O(N)$ in the worst case and requiring $O(\log N)$ time complexity. Worst-Case Weighted Fair Queueing (WF^{2}Q) [1] and its variants introduce eligible virtual times to provide stronger fairness guarantees than previous fair queueing approaches, limiting deviations in service error to $O(1)$. However, these algorithms still require at least $O(\log N)$ time complexity.

Because $O(\log N)$ time complexity scheduling is not good enough for high-speed links [3], there is great interest in developing lower complexity packet schedulers that still provide good fairness guarantees. Round-robin packet schedulers such as Deficit Round-Robin (DRR) [11] have been developed that only require $O(1)$ time complexity by servicing each flow for a continuous amount of time proportional to its weight. However, these round-robin schedulers deviate much more from GPS service than virtual time fair queueing approaches, with worst case service errors of $O(\phi_{max})$, where ϕ_{max} is the maximum flow weight. More recently, schedulers such as Smooth Round-Robin (SRR) [3] have been developed that in practice provide better fairness than previous round-robin approaches while retaining $O(1)$ time complexity. However, we show that the worst case service error of SRR is still worse than $O(N)$.

We present Group Round-Robin (GRR) scheduling, a novel scheduling framework that can be used with existing packet scheduling algorithms to improve the overall fairness and reduce the time complexity of scheduling competing flows. GRR introduces a novel flow group strategy that can be used to allow existing schedulers to consider groups of flows together instead of individually. By reducing the number of competing entities that exist-

ing schedulers must consider, this can improve both fairness and reduce time complexity when such properties are dependent on the number of entities scheduled. GRR's grouping strategy requires only simple queues for groups of flows, allowing it to be easily implemented in an efficient manner. GRR combines its grouping strategy with a specialized round-robin scheduling algorithm that utilizes the properties of GRR groups to schedule flows within groups in a manner that limits service error to $O(1)$ with only $O(1)$ time complexity. Under the practical assumption that GRR employs a small constant number of groups, we apply GRR to popular scheduling algorithms and show how GRR can be used to achieve constant bounds on fairness and time complexity for any of these algorithms.

This paper presents the design and analysis of GRR. Section II provides some background and discusses more precisely the notion of fairness. Section III describes the GRR scheduling algorithm. Section IV analyzes the fairness and time complexity of GRR. Section V describes how GRR can be applied to several existing scheduling algorithms to improve their fairness and time complexity bounds. The algorithms described include WFQ [5], SCFQ [6], SFQ [8], Hierarchical stride [13], WF^{2}Q [1], and SRR [3]. We also present and prove new results on the fairness bounds for several of these algorithms. Section VI discusses the delay bounds for GRR. Section VII presents some performance results from simulation studies to compare the fairness of existing scheduling algorithms with the same algorithms augmented with GRR. Finally, we present some concluding remarks.

II. Background

We first define the general terminology and state the assumptions we will use throughout the paper. Table I is a list of the general terminology we use. We denote Ψ as the server responsible for multiplexing an output link among a set of flows. Any scheduling algorithm can be viewed as an instance of Ψ. We only consider work-conserving servers, which use their maximum available bandwidth during any busy period. A busy period for Ψ is defined as a time interval during which there is at least one backlogged flow at any time. A flow is backlogged at time t for some server Ψ if it has bits left that are waiting to be sent, L(Q_i(t))>0. Since we are mainly concerned with busy periods, for any function $\Gamma(t_1, t_2)$, we will use the shorthand notation $\Gamma(t)$ to mean $\Gamma(t_0, t)$ for a busy period that started at t_0. For example, W(t)=$\int_{t_0}^{t} Rdt$.

To compare the performance of schedulers, we need to use a consistent measure of fairness. We describe two measures of fairness for this purpose. We start with the relative fairness of two flows since this is widely used as an indicator of fairness. A theorem in [6] states that if a packet-based algorithm guarantees $F_{i,j}(t_1, t_2) \leq \Delta_{i,j}$ for any interval (t_1, t_2) when S_i and S_j are backlogged, then $\Delta_{i,j}$ must be at least $\frac{1}{2}(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$. This result is used by proving that $\Delta_{i,j}$ is less than $\kappa(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$ for a scheduling algorithm, then making the claim that a small value of κ implies good fairness.

While relative fairness can be a useful measure, it is a weak measure compared to a measure of absolute service error which relates the work of a flow to the allocation the flow should have received given the work done by the scheduler. This service error reduces to a measure of service error versus GPS assuming that all flows are backlogged. To see that absolute service error is a stronger

TABLE I
General Terminology

Ψ	The server.
S_i	Flow i.
p_i^k	The k^{th} packet to have arrived for S_i.
$B(t)$	The set of backlogged flows at time t.
N	The number of backlogged flows in Ψ.
$a(p)$	The arrival time of packet p.
$d(p)$	The departure time of packet p.
$L_{i,max}$	The maximum size of a packet for S_i.
L_{max}	The maximum size of a packet in Ψ.
Q_i(t)	The queue of S_i at time t. $L(Q_i(t))$ denotes the number of bits on $Q_i(t)$.
ϕ_i	The weight assigned to S_i.
ϕ(t)	The sum of the weights of all flows backlogged at time t: $\sum_{S_j \in B(t)} \phi_j$.
$A_i(t_1, t_2)$	The total traffic that arrived for flow S_i.
R	The output link bandwidth of the server.
R_i	The guaranteed rate for flow S_i.
$W(t_1, t_2)$	The total traffic served by the server.
$w(t_1, t_2)$	The total normalized work: $\sum_{k=1}^{q} \frac{W(\tau_{k-1}, \tau_k)}{\sum_{S_i \in B(\tau_{k-1}+0)} \phi_i}$, where B($\tau_k + 0$) stays fixed during each interval (t_k, t_{k+1}) and $\tau_0 = t_1$, $\tau_q = t_2$.
$W_i(t_1, t_2)$	The amount of traffic served from S_i.
$w_i(t_1, t_2)$	The normalized work received by flow S_i: $\frac{W_i(t_1, t_2)}{\phi_i}$.
$F_{i,j}(t_1, t_2)$	The relative fairness of flows S_i and S_j: $\lvert w_i(t_1, t_2) - w_j(t_1, t_2) \rvert$.
$F_i(t_1, t_2)$	The absolute fairness: $\lvert w_i(t_1, t_2) - w(t_1, t_2) \rvert$.
$E_i(t_1, t_2)$	The absolute service error: $\phi_i F_i(t_1, t_2)$.

measure, suppose we have service error $E_i \leq \kappa L_{max}$, then the relative fairness is bounded as follows:

$$\begin{aligned} F_{i,j} &= |w_i - w_j| \leq |w_i - w| + |w_j - w| \\ &= \frac{E_i}{\phi_i} + \frac{E_j}{\phi_j} \leq \kappa(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j}) \end{aligned} \quad (1)$$

So an $O(1)$ bound on error always implies an $O(1)$ bound on relative fairness.

To see that relative fairness is a weaker fairness measure, we also show that an $O(1)$ bound on the relative fairness of an algorithm only translates to an $O(N)$ bound on the absolute service error. The service error for a flow S_i is $E_i(t_1, t_2) = |W_i(t_1, t_2) - \frac{\phi_i W(t_1,t_2)}{\phi(t)}| = |\frac{\phi_i w_i(t_1,t_2) \sum_{S_j \in B} \phi_j}{\sum_{S_j \in B} \phi_j} - \frac{\phi_i \sum_{S_j \in B} \phi_j w_j(t_1,t_2)}{\sum_{S_j \in B} \phi_j}| \leq \frac{\phi_i \sum_{S_j \in B} \phi_j |w_i(t_1,t_2) - w_j(t_1,t_2)|}{\sum_{S_j \in B} \phi_j} = \frac{\phi_i \sum_{S_j \in B} \phi_j F_{i,j}(t_1,t_2)}{\sum_{S_j \in B} \phi_j}$. If an algorithm guarantees $F_{i,j}(t_1, t_2) \leq \kappa(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$, then

$$\begin{aligned} E_i(t_1, t_2) &\leq \frac{\phi_i \sum_{S_j \in B} \phi_j \kappa(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})}{\sum_{S_j \in B} \phi_j} \\ &= \kappa L_{max}(1 + \frac{\phi_i}{\phi_{avg}}) \end{aligned} \quad (2)$$

where ϕ_{avg} is the average weight of all backlogged flows. Note that ϕ_i can get arbitrarily large compared to ϕ_{avg}. To show that the bounds on relative fairness, $\kappa(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$, cannot imply a service error bound of less than $O(N)$, consider an example of $N + 1$ flows, the first having weight N, and the rest having weights equal to 1. (2) would then result in an $O(N)$ service error bound. The SCFQ algorithm, for which the $O(1)$ relative fairness bound holds [6], has service error of $\frac{NL_{max}}{2}$ for this example. To put an upper bound on the error of algorithms with $\Delta_{i,j} = O(1)$, we can more formally argue based on (1) that

$$\begin{aligned} E_i &= \kappa L_{max}(1 + \frac{N\phi_i}{\phi(t)}) \\ &\leq \kappa L_{max}(1 + \frac{N\phi_i}{\phi_i}) = \kappa(N+1)L_{max} \end{aligned} \quad (3)$$

III. Group Round Robin Algorithm

GRR uses a novel grouping strategy to organize flows into groups of similar weight values which can be more easily scheduled. It then combines two scheduling algorithms: an intergroup scheduling algorithm to select a group from which to select a flow to service, and an intragroup scheduling algorithm to select a flow from within the selected group to service. Table II presents a list of GRR-specific terminology we use. The Group Round Robin algorithm can then be briefly described in three parts:

1) **Flow grouping strategy**: Flows are separated into groups of flows with similar weight values. Each group G is assigned flows with weight value between 2^{σ_G} to $2^{\sigma_G+1} - 1$, where $\sigma_G \geq 0$. Thus, a flow S_j is inserted into group G where $\sigma_G = \lfloor \log_2 \phi_j \rfloor$.
2) **Intergroup scheduling**: An existing scheduling algorithm is used to select a group from which to select a flow to service. A group is selected based on the group weight. Each group behaves like a composite flow with respect to the intergroup scheduler.
3) **Intragroup scheduling**: Once a group has been selected, a flow within the group is selected for service using a specialized round-robin algorithm.

The grouping strategy limits the number of groups that need to be scheduled since the number of groups grows at worst logarithmically with the largest flow weight value. If g is the number of groups, then $g \leq \lfloor \log_2 \phi_{max} \rfloor + 1$ where ϕ_{max} is the largest possible weight in the system. Even a very large 32-bit flow weight would limit the number of groups to no more than 32. As a result, the intergroup scheduler never needs to schedule a large number of groups which limits the impact of skewed weight distributions on groups. The grouping strategy also limits the weight distributions that the intragroup scheduler needs to consider since the range of weight values within a group is less than a factor of two. As a result, the intragroup scheduler never needs to schedule flows with skewed weight distributions since the flows within a group must have relatively similar weight values.

While GRR is designed to leverage existing scheduling algorithms for intergroup scheduling, GRR takes advantage of its grouping strategy by using a specialized deficit round-robin algorithm for intragroup scheduling to provide good fairness and time complexity bounds. Compared to DRR, the GRR intragroup scheduler has two important differences. First, all flow weights in a group G are normalized with respect to the minimum possible weight, $\phi_{min} = 2^{\sigma_G}$, for any flow in the group. Second, since normalized flow weights may be non-integers, GRR can provide fractional packet allocations that are accumulated as part of a flow's deficit.

The GRR intragroup algorithm considers the scheduling of flows in rounds. A *round* is one pass through a group's queue of flows from beginning to end. The group queue of flows does not need to be sorted in any manner. During each round, the GRR intragroup algorithm considers the flows in round-robin order. For each

TABLE II
GRR TERMINOLOGY

g	The number of groups.
$G(i)$	The group to which S_i belongs.
σ_G	The order of G.
B_G(t)	The set of backlogged flows in group G.
ϕ_G(t)	The group weight: $\sum_{S_i \in B_G(t)} \phi_i$.
R_G	The guaranteed rate for group G.
$W_G(t_1, t_2)$	The amount of traffic served for group G: $\sum_{S_i \in G} W_i(t_1, t_2)$.
$w_G(t_1, t_2)$	The normalized traffic for group G: $\sum_{k=1}^{q} \frac{W_G(\tau_1, \tau_2)}{\sum_{S_i \in B_G(\tau_{k-1}+0)} \phi_i}$ where $B_G(\tau_k + 0)$ stays fixed during each interval (t_k, t_{k+1}) and $\tau_0 = t_1$, $\tau_q = t_2$.
$F_{G,H}(t_1, t_2)$	The relative fairness of groups G and H: $\lvert w_G(t_1, t_2) - w_H(t_1, t_2) \rvert$.
$F_G(t_1, t_2)$	The absolute fairness of group G: $\lvert w_G(t_1, t_2) - w(t_1, t_2) \rvert$.
$F_{i,G}(t_1, t_2)$	The group relative fairness of flow S_i: $\lvert w_i(t_1, t_2) - w_G(t_1, t_2) \rvert$.
$E_{i,G}(t_1, t_2)$	The group relative error of flow S_i: $\phi_i F_{i,G}(t_1, t_2)$.
D_i(t)	The deficit of S_i at time t.

backlogged flow S_i, the scheduler serves a maximum of $(\frac{\phi_i}{\phi_{min}})L_{max} + D_i(r-1)$ bits. $D_i(r)$, the deficit of flow S_i after round r, is defined recursively as $D_i(r) = (\frac{\phi_i}{\phi_{min}})L_{max} + D_i(r-1) - \sum_{k=k_i^{r-1}+1}^{k_i^r} L(p_i^k)$, where k_i^r is the number of S_i packets that left the server up to and including round r, with $D_i(0) = 0$. Thus, in each round, S_i is allotted $(\frac{\phi_i}{\phi_{min}})L_{max}$ bits plus any additional left-over from the previous round, and $D_i(r)$ keeps track of the amount of service that S_i missed because the packet at the head of the flow's queue was too large to fit in the remaining allotted space. We observe that $D_i(r) < L_{i,max}$ after any round r.

Consider the following example to illustrate further how GRR scheduling works. Suppose we use WFQ for the intergroup scheduler in the GRR framework. Consider a set of six flows that need to be scheduled, one flow S_1 with weight 12, two flows S_2 and S_3 each with weight 3, and the other three flows S_4, S_5, and S_6 each with weight 2. Assume that all flows are backlogged and the size of all the packets is $L_{max} = 1$ for simplicity. The six flows will be put into two groups G_1 and G_2 as follows: $B_{G_1} = \{S_2, S_3, S_4, S_5, S_6\}$ and $B_{G_2} = \{S_1\}$. The weight of the groups are $\sigma_1 = 12$ and $\sigma_2 = 12$. WFQ will consider the groups in this order: $G_1, G_2, G_1, G_2, G_1, G_2, G_1, G_2, G_1, G_2, G_1, G_2$. G_2 will schedule flow S_1 every time G_2 is considered for service since it has only one flow. We will show the order of service for the first two rounds of G_1. Rounds 3 and 4 of G_1 has the same order of service as rounds 1 and 2. In beginning of round 1 in G_1, each flow starts with 0 deficit and gains $\frac{\phi_i}{\phi_{min(G_1)}}$ and $\phi_{\min(G_1)} = 2$. The maximum service that flows S_2, S_3, S_4, S_5, and S_6 can receive in round 1 are 1.5, 1.5, 1, 1, and 1, respectively. Since packets have to be transmitted as a unit, the scheduler will serve 1 packet from each flow in G_1 during round 1. After the first round, the deficit for S_2, S_3, S_4, S_5, and S_6 are 0.5, 0.5, 0, 0, and 0. In the beginning of round 2, each flow gets another $\frac{\phi_i}{\phi_{min}}$ allocation, and the maximum allowed service for S_2, S_3, S_4, S_5, and S_6 becomes 2, 2, 1, 1, and 1. S_2 sends two packets, followed by two packets from S_3 and one packet each from S_4, S_5 and S_6. After round 2, the deficit of all the flows in G_2 becomes 0. The sequence of packets that the scheduler serves is $\{p_2^1, p_1^1, p_3^1, p_1^2, p_4^1, p_1^3, p_5^1, p_1^4, p_6^1, p_1^5, p_2^2, p_1^6, p_2^3, p_1^7, p_3^2, p_1^8, p_3^3, p_1^9, p_4^2, p_1^{10}, p_5^2, p_1^{11}, p_6^2, p_1^{12}\}$, where p_i^k is the k^{th} packet to have arrived for flow S_i.

IV. GRR FAIRNESS AND TIME COMPLEXITY

We first analyze the fairness bounds of GRR and then discuss its time complexity. GRR limits service error by dividing up the scheduling problem into intragroup scheduling and intergroup scheduling. We start with the relative fairness of the intragroup algorithm and show that the algorithm provides $O(1)$ relative fairness between any two flows. We present the following lemma.

Lemma 1: The relative fairness $F_{i,j}$ of the GRR intragroup round-robin scheduling algorithm between any two flows S_i and S_j is bounded as follows: $F_{i,j} \leq 2(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$.

Proof: For any interval (t_1, t_2) when S_i is continuously backlogged, let r_1^i be the round before the round when S_i is first considered after time t_1, and r_2^i the last round before time t_2 when S_i sends. Then S_i receives during (t_1, t_2) a total service equal to $(r_2^i - r_1^i)(\frac{\phi_i}{\phi_{min}})L_{max} + D_i(r_1^i) - D_i(r_2^i)$. Let S_j be another session continuously backlogged in the interval (t_1, t_2), and let us assume without loss of generality that S_j is after S_i in the group. Then $r_1^i - 1 \leq r_1^j \leq r_1^i$ and $r_2^i - 1 \leq r_2^j \leq r_2^i$. Subtracting, we get $r_2^i - r_1^i - 1 \leq r_2^j - r_1^j \leq r_2^i - r_1^i + 1$ or $\lvert (r_2^i - r_1^i) - (r_2^j - r_1^j) \rvert \leq 1$. We also observe that $\lvert D_j(r_1^j) - D_j(r_2^j) \rvert \leq L_{max}$ for any j, since $D_j(t) < L_{max}$ at any time t. It then follows that the relative fairness $F_{i,j}$ between flows S_i and S_j is: $F_{i,j} = \lvert w_i(t_1, t_2) - w_j(t_1, t_2) \rvert = \lvert \{(r_2^i - r_1^i)(\frac{L_{max}}{\phi_{min}}) + \frac{(D_i(r_1^i) - D_i(r_2^i))}{\phi_i} \} -$

$\{(r_2^j - r_1^j)(\frac{L_{max}}{\phi_{min}}) + \frac{(D_j(r_1^j)-D_j(r_2^j))}{\phi_j}\}| \leq |\{(r_2^i - r_1^i) - (r_2^j - r_1^j)\}(\frac{L_{max}}{\phi_{min}})| + |\frac{(D_i(r_1^i)-D_i(r_2^i))}{\phi_i}| + |\frac{(D_j(r_1^j)-D_j(r_2^j))}{\phi_j}| \leq \frac{L_{max}}{\phi_{min}} + \frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j}$. Because $\phi_{min} \leq \phi_i, \phi_j < 2\phi_{min}$, we can simplify this to the looser bounds $F_{i,j} \leq \frac{3L_{max}}{\phi_{min}}$ or $F_{i,j} \leq 2(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$. ■

Note that if we only consider full rounds, then the bounds on $F_{i,j}$ would be only $\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j}$. In all cases, the relative fairness bounds are $O(1)$.

We now consider the absolute service error of the intragroup algorithm and show a stronger result, namely that the algorithm provides $O(1)$ absolute service error. We present the following lemma.

Lemma 2: The absolute service error $E_{i,G}$ of the GRR intragroup round-robin scheduling algorithm for flow S_i is bounded as follows: $E_{i,G} \leq 5L_{max}$.

Proof: Consider the group relative fairness $F_{i,G}$ of a flow $S_i \in G$: $F_{i,G}(T_0, T) = |w_i(T_0, T) - w_G(T_0, T)|$. To simplify the analysis, we will first consider $F_{i,G}$ in between rounds. Let t_k be the time round k finishes, and let $t_0 = T_0$ and $t_r = T$. Then during round k, W_j increases by $(\frac{\phi_j}{\phi_{min}})L_{max} + D_j(t_{k-1}) - D_j(t_k)$ for any flow S_j in G that is backlogged during that round. We can then write $w_G(t_{k-1}, t_k) = \frac{\sum_{S_j \in B_G(t_{k-1}+0)} W_j(t_{k-1}, t_k)}{\sum_{S_j \in B_G(t_{k-1}+0)} \phi_j} = \frac{\sum_{S_j \in B_G(t_{k-1}+0)} \{(\frac{\phi_j}{\phi_{min}})L_{max} + D_j(t_{k-1}) - D_j(t_k)\}}{\phi_G(t_{k-1}+0)} = \frac{L_{max}}{\phi_{min}} + \frac{D_G(t_{k-1}+0) - D_G(t_k-0)}{\phi_G(t_{k-1}+0)}$, where we denote by $D_G(t)$ the sum of the deficits of the backlogged flows in G at time t. We make the distinction between $D_G(t_k - 0)$ and $D_G(t_k + 0)$ since after a round, we may adjust $D_G(t)$ by assigning some initial deficit to a new flow that arrived during that round. We can have three situations after each round:

1) The backlog set B_G remains unchanged (no session arrived or became idle). Then $D_G(t_k - 0) = D_G(t_k + 0)$ and $\phi_G(t_{k-1} + 0) = \phi_G(t_k + 0)$.
2) A new session S_{new} arrived: $\phi_G(t_k + 0) = \phi_G(t_{k-1} + 0) + \phi_{new}$. We place the new session at the beginning of the queue (so that it does not run during the current round) and after the round, we assign it a deficit of $\frac{\phi_{new} D_G(t_k-0)}{\phi_G(t_{k-1}+0)}$. Then, $D_G(t_k + 0) = D_G(t_k - 0) + \frac{\phi_{new} D_G(t_k-0)}{\phi_G(t_{k-1}+0)} = \frac{D_G(t_k-0)\phi_G(t_k+0)}{\phi_G(t_{k-1}+0)}$. Thus, $\frac{D_G(t_k-0)}{\phi_G(t_{k-1}+0)} = \frac{D_G(t_k+0)}{\phi_G(t_k+0)}$.
3) A session S_{dep} departed (became idle): $\phi_G(t_k + 0) = \phi_G(t_{k-1} + 0) - \phi_{dep}$. In this case, the work that the departing flow executes during the round is less than its quota. To keep the analysis simple, we can assume for the purpose of keeping track of fairness that the departing flow has received $\phi_{dep}(\frac{L_{max}}{\phi_{min}} - \frac{D_G(t_k+0)}{\phi_G(t_k+0)})$ which is less than $\frac{\phi_{dep} L_{max}}{\phi_{min}}$ and positive (since $\frac{D_G(t_k+0)}{\phi_G(t_k+0)} \leq \frac{|B_G(t_k+0)|L_{max}}{|B_G(t_k+0)|\phi_{min}} = \frac{L_{max}}{\phi_{min}}$). Then the normalized work done for round k is $\frac{L_{max}}{\phi_{min}} + \frac{D_G(t_{k-1}+0) - (D_G(t_k-0) + \frac{\phi_{dep} D_G(t_k+0)}{\phi_G(t_k+0)})}{\phi_G(t_{k-1}+0)}$. Since we make no changes to D_G after round k, $D_G(t_k+0) = D_G(t_k-0)$ and we have a normalized service of $\frac{L_{max}}{\phi_{min}} + \frac{D_G(t_{k-1}+0) \frac{\phi_G(t_k+0) D_G(t_k+0)}{\phi_G(t_k+0)}}{\phi_G(t_{k-1}+0)} = \frac{L_{max}}{\phi_{min}} + \frac{D_G(t_{k-1}+0)}{\phi_G(t_{k-1}+0)} - \frac{D_G(t_k+0)}{\phi_G(t_k+0)}$.

Of course, we can have any number of sessions arrive and depart during the same round, in which case the results in situations 2 and/or 3 still apply by superposition. Thus, in all three cases, the normalized work during a round can be written as $\frac{L_{max}}{\phi_{min}} + \frac{D_G(t_{k-1}+0)}{\phi_G(t_{k-1}+0)} - \frac{D_G(t_k+0)}{\phi_G(t_k+0)}$, so $F_{i,G}(T_0, T) = |\sum_{k=1}^{r}(\frac{1}{\phi_i}[\frac{L_{max}\phi_i}{\phi_{min}} + D_i(t_{k-1}) - D_i(t_k)]) - \sum_{k=1}^{r}[\frac{L_{max}}{\phi_{min}} + \frac{D_G(t_{k-1}+0)}{\phi_G(t_{k-1}+0)} - \frac{D_G(t_k+0)}{\phi_G(t_k+0)}]| = |\frac{D_i(t_0)-D_i(t_r)}{\phi_i} - \frac{D_G(t_0+0)}{\phi_G(t_0+0)} + \frac{D_G(t_r+0)}{\phi_G(t_r+0)}|$. We now make use of the fact that $0 \leq D_G(t) \leq |B_G(t)|L_{max} \leq (\frac{\phi_G(t)}{\phi_{min}})L_{max}$ and $0 \leq D_i \leq L_{max}$ to get $F_{i,G}(T_0, T) \leq |\frac{D_i(T_0)-D_i(T)}{\phi_i}| + |\frac{D_G(T_0)}{\phi_G(T_0)} - \frac{D_G(T)}{\phi_G(T)}| \leq \frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_{min}} \leq \frac{L_{max}}{\phi_i} + \frac{2L_{max}}{\phi_i} = \frac{3L_{max}}{\phi_i}$ since $\phi_{min} > \frac{\phi_i}{2}$. The group relative error $E_{i,G}$ in between rounds is then bounded by $E_{i,G}(T_0, T) \leq 3L_{max}$. Since during a round, a flow's normalized work w_i can get ahead or behind by at most an additional $\frac{L_{max}}{\phi_{min}}$, in general $F_{i,G}(T_0, T) \leq \frac{5L_{max}}{\phi_i}$ and $E_{i,G}(T_0, T) \leq 5L_{max}$ for any time interval (T_0, T). ■

Given the $O(1)$ relative fairness and absolute service error bounds for the GRR intragroup scheduling algorithm, we can now analyze the overall GRR algorithm. We first show the following theorem that states the GRR algorithm relative fairness is bounded by the relative fairness of the intergroup scheduler plus a constant factor.

Theorem 1: The relative fairness $F_{i,j}$ of the GRR scheduling algorithm between any two flows S_i and S_j is bounded by a constant plus the relative fairness between the respective groups containing the flows as follows: $F_{i,j} \leq \frac{5L_{max}}{\phi_i} + \frac{5L_{max}}{\phi_j} + F_{G(i),G(j)}$.

Proof: For any two flows S_i and S_j in groups G(i) and G(j) and continuously backlogged in the interval (t_1, t_2), we have: $F_{i,j}(t_1, t_2) = |w_i(t_1, t_2) - w_j(t_1, t_2)| = |w_i(t_1, t_2) - w_{G(i)}(t_1, t_2) + w_{G(i)}(t_1, t_2) - w_{G(j)}(t_1, t_2) + w_{G(j)}(t_1, t_2) - w_j(t_1, t_2)| \leq |w_i(t_1, t_2) - w_{G(i)}(t_1, t_2)| + |w_{G(j)}(t_1, t_2) - w_j(t_1, t_2)| + |w_{G(i)}(t_1, t_2) - w_{G(j)}(t_1, t_2)| \leq F_{i,G(i)} + F_{j,G(j)} + F_{G(i),G(j)}$. Since the intragroup relative fairness between

two flows is bounded by Lemma 1, it follows that the relative fairness of the overall GRR algorithm between any two flows is bounded by $F_{i,j}(t_1,t_2) \leq \frac{5L_{max}}{\phi_i} + \frac{5L_{max}}{\phi_j} + F_{G(i),G(j)}$. Of course, if both flows are part of the same group, then $F_{i,j} \leq \frac{2L_{max}}{\phi_i} + \frac{2L_{max}}{\phi_j}$. This is a special case, but one that would occur frequently in practice, where many flows tend to have the same or similar weights. ■

In a similar manner, we can show the following theorem which states the stronger result that the overall GRR algorithm service error is bounded by the service error of the intergroup scheduler plus a constant factor.

Theorem 2: The absolute service error E_i of the GRR scheduling algorithm of flow S_i is bounded by a constant plus the absolute service error of the intergroup scheduling algorithm as follows: $E_i \leq 5L_{max} + E_G$.

Proof: Let flow S_i be backlogged in group G during the interval (T_0,T). Then $F_i(T_0,T) = |w_i(T_0,T) - w(T_0,T)| = |w_i(T_0,T) - w_G(T_0,T) + w_G(T_0,T) - w(T_0,T)| \leq |w_i(T_0,T) - w_G(T_0,T)| + |w_G(T_0,T) - w(T_0,T)| = F_{i,G}(T_0,T) + F_G(T_0,T)$. The overall error will be at most $E_i(T_0,T) = \phi_i F_i(T_0,T) \leq E_{i,G}(T_0,T) + \phi_i F_G(T_0,T) = E_{i,G}(T_0,T) + (\frac{\phi_i}{\phi_G})E_G(T_0,T) \leq E_{i,G}(T_0,T) + E_G(T_0,T)$. We have shown that $E_{i,G} \leq 5L_{max}$ for the intragroup round robin scheduler. Thus, the overall error will be on the order of the overall error of the intergroup scheduler used: $E_i(T_0,T) \leq 5L_{max} + E_G(T_0,T)$. ■

We now analyze GRR time complexity. Using GRR, scheduling the next packet to transmit entails choosing the group and picking the appropriate flow from within the group that gets to send a packet. The time for the GRR intragroup scheduler to select a flow for service from a group is $O(1)$. This follows from the fact that the round robin always considers backlogged flows in the same order, and serves each with at least one packet since $L(p_i^k) \leq L_{max}$ and $\frac{\phi_i}{\phi_{min}} \geq 1$. Non-backlogged flows are logically removed from the queues so that the scheduler does not waste any time looping through flows with no packets to send. Since selecting a flow is $O(1)$ for the intragroup round-robin algorithm, the overall complexity will be the same as that of the intergroup scheduler.

V. Using GRR with Other Scheduling Algorithms

We describe how GRR can be used with a number of existing scheduling algorithms by incorporating those algorithms as the GRR intergroup scheduling algorithm. We show how using GRR in conjunction with these algorithms can improve their fairness and service error bounds and reduce their time complexity. Applying GRR is simple, provided that the intergroup algorithms handle dynamically changing weights properly. This is true for all the six popular schedulers we consider in this context: WFQ, SCFQ, SFQ, hierarchical stride (HS), WF^{2}Q, and SRR. We discuss the relative fairness and absolute service error bounds for each of these algorithms, presenting for the first time consistent bounds using the same fairness and service error measures for all of these algorithms. The more extensive proofs for these results are included in the Appendix. We combine these algorithms with GRR to construct six new scheduling algorithms: GWFQ, GSCFQ, GSFQ, GHS, GWF^{2}Q, and GSRR.

A. GWFQ

WFQ [5], [10] is a virtual time fair queueing algorithm. WFQ emulates GPS by serving packets in the order in which they would finish service under GPS. WFQ introduced virtual finishing times (VFT) for this purpose and services the flow with the packet with the earliest VFT.

For WFQ, the relative fairness between any two flows S_i and S_j is bounded by $\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j}$ as shown by the proof for Lemma 3 in the Appendix. (3) then implies that the absolute service error of WFQ is bounded by $(N+1)L_{max}$, where N is the number of flows being scheduled. We can see that the service error bound is not less than $O(N)$ by considering the example of $N+1$ flows, the first with weight N and the rest weight 1. If all packets are of size L_{max}, WFQ will service N packets from the first flow before servicing any other flows, resulting in service error of $\frac{NL_{max}}{2}$. Because WFQ must order the flows based on their VFTs, the time complexity of scheduling is $O(\log N)$.

We can combine WFQ with GRR to derive a new scheduling algorithm GWFQ that has both a lower service error bound and lower time complexity. GWFQ is simply the GRR scheduler using WFQ as the intergroup scheduling algorithm. GWFQ only uses WFQ for scheduling among groups, reducing the number of entities that are considered by the WFQ algorithm. This reduces service error and time complexity since the service error bound and time complexity of WFQ grow with the number of entities being scheduled.

For GWFQ, the relative fairness between any two flows S_i and S_j is bounded by $6(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$, as computed based on Theorem 1. This is an upper bound but may not be a tight bound. Theorem 2 shows that the absolute service error is a constant factor plus the absolute service error of the intergroup scheduler. Since the WFQ intergroup scheduler has absolute service error $(g+1)L_{max}$,

where g is the number of groups, the absolute service error of GWFQ is $(g+1)L_{max}+5L_{max}$, which is $O(g)$. Similarly, the time complexity of GWFQ is $O(\log g)$. If we assume that the number of groups is bounded by a constant, this reduces to a $O(1)$ bound on absolute service error and time complexity.

B. GSCFQ

SCFQ is another virtual time scheduling algorithm. For SCFQ, the relative fairness between any two flows S_i and S_j has been shown to be bounded by $\frac{L_{max}}{\phi_i}+\frac{L_{max}}{\phi_j}$ [6]. [6] did not show a service error bound, but as in the case of WFQ, (3) implies the absolute service error bound of SCFQ is $(N+1)L_{max}$, where N is the number of flows being scheduled. The time complexity of SCFQ scheduling is $O(\log N)$. The basic fairness and time complexity properties of SCFQ are similar to WFQ.

Like GWFQ, we can use SCFQ as the intergroup scheduling algorithm with GRR to derive a new scheduling algorithm GSCFQ that has both a lower service error bound and lower time complexity. Since SCFQ and WFQ have the same relative fairness and service error bounds, GSCFQ has the same relative fairness and service error bounds as GWFQ, namely relative fairness bounded by $6(\frac{L_{max}}{\phi_i}+\frac{L_{max}}{\phi_j})$ and absolute service error of $(g+1)L_{max}+5L_{max}$. Similarly, the time complexity of GSCFQ is $O(\log g)$.

C. GSFQ

SFQ is another virtual time scheduling algorithm. For SFQ, the relative fairness between any two flows S_i and S_j has been shown to be bounded by $\frac{L_{max}}{\phi_i}+\frac{L_{max}}{\phi_j}$ [8]. [8] did not show a service error bound, but as in the case of WFQ and SCFQ, (3) implies the absolute service error bound of SCFQ is $(N+1)L_{max}$, where N is the number of flows being scheduled. The time complexity of SFQ scheduling is $O(\log N)$.

Like GWFQ and GSCFQ, we can use SFQ as the intergroup scheduling algorithm with GRR to derive a new scheduler GSFQ that has both a lower service error bound and lower time complexity. Since SFQ and WFQ have the same relative fairness and service error bounds, GSFQ has the same relative fairness and service error bounds as GWFQ, namely relative fairness bounded by $6(\frac{L_{max}}{\phi_i}+\frac{L_{max}}{\phi_j})$ and absolute service error of $(g+1)L_{max}+5L_{max}$. Similarly, the time complexity of GSFQ is $O(\log g)$.

D. GHS

A different group approach than that of GRR was previously proposed in the context of stride CPU scheduling [13]. We discuss an adaptation of that algorithm for packet scheduling that we refer to as HS. HS arranges flows in a balanced binary tree, where the flows are the leaves, and any internal node behaves with respect to its parent like a flow with a weight equal to the sum of the weights of its children. When an internal node is selected, then it in turn selects one of its children, until a leaf node is reached and serviced. Normalized work counters for the flow and all of its ancestors are incremented with the amount of bits transmitted by the flow. A parent node selects from its children p and q the node whose normalized work is less than the normalized work of the parent. If $w_p = w_q = w$, then some tie-breaking policy can be used, for example, select the node with the higher weight, or select the left node, etc. The benefit of this hierarchical approach can be most easily illustrated by the case of $N+1$ flows, the first with weight N and the rest weight 1. If all packets are of size L_{max}, a non-hierarchical stride scheduler would service N packets from the first flow before servicing any other flows, resulting in $O(N)$ service error. However, HS aggregates multiple flows so that it ends up interleaving the execution of the weight N flow with the other flows of weight 1, resulting in a smaller $O(\log N)$ service error.

For HS, the relative fairness between any two flows S_i and S_j is bounded by $\frac{\lceil \log(N) \rceil L_{max}}{\min(\phi_i,\phi_j)}$ where N is the number of flows being scheduled, according to Lemma 5. The absolute service error of HS is bounded by $\lceil \log(N) \rceil L_{max}$ as stated in Lemma 4. Both lemmas are prove in the Appendix. Since HS must do a traversal of the balanced binary tree from the root to a leaf, the time complexity of HS scheduling is $O(\log N)$.

The hierarchical grouping strategy of HS provides some of the same benefits of the GRR framework, though it requires more complex data structures. However, we can combine HS with GRR to derive a new scheduling algorithm GHS that provides even lower service error and time complexity. GHS is simply the GRR scheduler using HS as the intergroup scheduling algorithm. GHS only uses HS for scheduling among groups, reducing the number of entities that are considered by the HS algorithm. This reduces service error and time complexity since the service error bound and time complexity of HS grow with the number of entities being scheduled.

For GHS, the relative fairness between any two flows S_i and S_j is $F_{i,j} \leq F_{G(i),G(j)} + \frac{5L_{max}}{\phi_i} + \frac{5L_{max}}{\phi_j} \leq \frac{(\lceil \log g \rceil)L_{max}}{\min(\phi_{G(i)},\phi_{G(j)})} + \frac{10L_{max}}{\min(\phi_i,\phi_j)} \leq \frac{L_{max}(\lceil \log g \rceil+10)}{\min(\phi_i,\phi_j)}$, where g is the number of groups of flows being scheduled. The absolute service error of GHS is $E_i \leq E_{i,G(i)} + E_{G(i)} \leq 5L_{max} + (\lceil \log g \rceil)L_{max} = (\lceil \log g \rceil + 5)L_{max}$. The time

complexity of GHS is $O(1) + O(\log g) = O(\log g)$.

Note that the motivation of hierarchy in HS to reduce GPS service error bounds is different from the conventional notion of hierarchical packet scheduling algorithms which seek to provide H-GPS fairness [2]. The idea of grouping flows together is also at the heart of these algorithms, but their goal is to emulate H-GPS instead of GPS. The aim is to provide isolation in link sharing and to implement different policy-based service classes. Thus, fairness is provided among the children groups of a node, but not across the entire system, and unused service from a group is distributed solely inside the parent group. However, the idea of using an instance of the same type of virtual time server at each node can also be adapted to simulate GPS, by letting the weights of the group nodes vary when flows enter or leave the groups. Still, such an approach gives error and computational complexity bounds that are a factor of k larger than those for the individual server, where k is the height of the tree, and are not well-suited for our purpose.

E. GWF^{2}Q

WF^{2}Q is a virtual time algorithm that is identical to WFQ, except it considers for service only the packets that would have already started service under the equivalent GPS. This difference though enables WF^{2}Q to provide the lowest service error of all fair queueing algorithms. For WF^{2}Q, the absolute service error has been shown to be bounded by L_{max} [1], which is $O(1)$. As a result, (1) implies for WF^{2}Q that its relative fairness between any two flows S_i and S_j is bounded by $(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$. The time complexity of WF^{2}Q scheduling is $O(\log N)$.

We can combine WF^{2}Q with GRR to derive a new scheduling algorithm GWF^{2}Q that preserves the $O(1)$ service error bound while providing lower time complexity. GWF^{2}Q is simply the GRR scheduler using WF^{2}Q as the intergroup scheduling algorithm. GWF^{2}Q only uses WF^{2}Q for scheduling among groups, reducing the number of entities that are considered by the WF^{2}Q algorithm. This reduces time complexity since the time complexity of WF^{2}Q grows with the number of entities being scheduled.

For GWF^{2}Q, the relative fairness between any two flows S_i and S_j is bounded by $6(\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j})$, as computed based on Theorem 1. Theorem 2 shows that the absolute service error is a constant factor plus the absolute service error of the intergroup scheduler. Since the WF^{2}Q intergroup scheduler has absolute service error L_{max}, the absolute service error of GWF^{2}Q is $6L_{max}$, which is $O(1)$. Similarly, the time complexity of GWF^{2}Q is $O(\log g)$. If we assume that the number of groups is constant, this reduces to a $O(1)$ bound on absolute service error and time complexity.

F. GSRR

SRR is a scheduling algorithm that services flows in a fixed order similar to round-robin scheduling. SRR introduces a Weight Matrix and uses the concept of a k-order Weight Spread Sequence (WSS), where k is the number of bits needed to store the weight of the flows. The Weight Matrix consists of binary vectors coded from the weights of the flows. SRR then scans the elements of the Weight Matrix in a fixed order specified by WSS and selects the flow to execute whose weight corresponds to the matrix element selected.

For SRR, the relative fairness between any two flows S_i and S_j has been shown to be bounded by $\frac{(k+2)L_{max}}{2\min(\phi_i,\phi_j)}$ [3]. The absolute service error of SRR is $E_i \leq \frac{k(N+1)}{2} = O(Nk)$, as stated in Lemma 6, which we prove in the Appendix . The time complexity of SRR scheduling is $O(1)$.

We can combine SRR with GRR to derive a new scheduling algorithm GSRR that can preserve the $O(1)$ time complexity of SRR while providing a lower service error bound. GSRR is simply the GRR scheduler using SRR as the intergroup scheduling algorithm. However, since the weight of a group increases with the number of flows, we cannot have a bound for k as assumed in SRR, so we cannot use a pre-computed WSS. We can have SRR simulate in real time the WSS, allowing theoretically for any size weights, at the expense of computational complexity. In the Appendix, we present a method of dynamically generating the WSS that preserves the $O(1)$ time complexity of SRR and improves the space complexity from $O(2^k)$ to $O(k)$, while also making the algorithm more scalable for our purpose.

For GSRR, the relative fairness between any two flows S_i and S_j is bounded by $\frac{(k+22)L_{max}}{2\min_{S_p \in (G(i) \cup G(j))}(\phi_p)}$, where $k = \log_2(\phi_{G,max})$ and $\phi_{G,max}$ is the largest group weight. In worst case, k is $O(\log N)$. The absolute service error of SRR is $E_i \leq 5L_{max} + \frac{gk}{2}L_{max} = O(g \log N)$. Applying the grouping strategy to SRR thus reduces the error from $O(kN)$ to $O(g \log N)$ where $g = k = \log \phi_{max}$, while maintaining the $O(1)$ time complexity.

VI. Delay Bounds for GRR

Even though the main goal of GRR is to improve fairness and time complexity, we can also show reasonable delay bounds that make GRR well-suited for guaranteeing QoS in high-speed networks. Since we cannot predict network usage in the future, we give our delay bounds

without making any assumption about the incoming traffic envelope, or other flow control characteristics.

We will show that GRR belongs to the GR (guaranteed rate) class of algorithms [7] by showing that for any packet p_i^k that arrives for flow S_i, $d(p_i^k) - GRC(p_i^k) \leq \beta_i^{GRR}$, where β_i^{GRR} is constant. GRC, the guaranteed rate clock, is defined in [7] and is similar to the concept of expected arrival time used in [8]. $GRC(p_i^k) = \max(a(p_i^k), GRC(p_i^{k-1})) + \frac{L(p_i^k)}{R_i}$ and $GRC(p_i^0) = 0$. We observe that $GRC(p_i^k) \geq \frac{A_i(a(p_i^1), a(p_i^k))}{R_i}$, with equality if all flows are continuously backlogged under GPS. Also, we note that $GRC(p_i^k) \geq d^{GPS}(p_i^k)$ [7].

We also use the concept of a WFI as defined in [2]: a server guarantees for S_i a WFI of α_i if for any packet p_i^k, the following holds $W_i(t_1, d(p_i^k)) \geq (\frac{\phi_i}{\phi})W(t_1, d(p_i^k)) - \alpha_i$ for any t_1 such that S_i is continuously backlogged during $(t_1, d(p_i^k))$.

Let us consider the time interval (a, d) where $a = a(p_i^1)$ and $d = d(p_i^k)$. Since $L(Q_i(a-0)) = 0$, the work of S_i, $W_i(a, d)$ equals the total length of the first k packets arrived at the flow's queue, $A_i(a, a(p_i^k)) \leq R_i GRC(p_i^k)$. On the other hand, we have $W_i(a, d) \geq (\frac{\phi_i}{\phi_{G(i)}})W_{G(i)}(a, d) - 5L_{max}$ due to the error bounds proven for the intragroup round-robin. Also, $W_{G(i)}(a, d) \geq (\frac{\phi_{G(i)}}{\phi})W(a, d) - \alpha_{G(i)}$, where $\alpha_{G(i)}$ is the group's WFI in the intergroup algorithm. Since the server is busy during (a, d) and work-conserving, we have $W = R(d-a)$. Combining the above relations, we obtain $R_i GRC(p_i^k) \geq W_i(a, d) \geq (\frac{\phi_i}{\phi_{G(i)}})((\frac{\phi_{G(i)}}{\phi})W(a, d) - \alpha_{G(i)}) - 5L_{max} = (\frac{\phi_i}{\phi})R(d-a) - (\frac{\phi_i}{\phi_{G(i)}})\alpha_{G(i)} - 5L_{max}$. Hence, $R_i(d - GRC(p_i^k)) \leq 5L_{max} + (\frac{\phi_i}{\phi_{G(i)}})\alpha_{G(i)}$, or $d - GRC(p_i^k) \leq \frac{5L_{max}}{R_i} + \frac{\alpha_{G(i)}\phi_i}{R_i\phi_{G(i)}}$. Thus, $\beta_i^{GRR} = \frac{5L_{max}}{R_i} + \frac{\alpha_{G(i)}\phi_i}{R_i\phi_{G(i)}}$.

For WF^{2}Q and WF^{2}Q$^+$, α_i is $L_{i,max} + \frac{(L_{max} - L_{i,max})\phi_i}{\phi}$. Therefore, $\beta_i^{GWF^2Q}$ can be bounded by $\frac{5L_{max}}{R_i} + \frac{L_{G(i),max}}{R_i} + (\frac{\phi_i}{\phi})\frac{(L_{max} - L_{G(i),max})}{R_i}$.

For WFQ, SCFQ, and SFQ, $\beta_i^{WFQ} = \frac{L_{max}}{R}$ [7], $\beta_i^{SCFQ} = \sum_{S_j \in B, j \neq i} \frac{L_{j,max}}{R}$ [7], and $\beta_i^{SFQ} = \sum_{S_j \in B} \frac{L_{j,max}}{R} - \frac{L(p_i^k)}{R_i}$ [8]. However, WFQ, SCFQ, and SFQ all have α_i that can grow as $O(N)$, where N is the number of entities scheduled [2]. We can still bound the WFI by observing that for any algorithm, $\text{WFI}_i \leq E_i$. This allows us to bound β_i^{GRR} for WFQ, SCFQ, and SFQ by $\frac{5L_{max}}{R_i} + (\frac{\phi_i}{\phi_G})(g+1)\frac{L_{max}}{R_i}$. We see that compared to the β_i bounds for the standalone WFQ, SCFQ, and SFQ servers, β_i^{GRR} increases as $\frac{(g+1)}{R_i}$. The $\frac{1}{R_i}$ factor is due to the round robin, and algorithms such as SRR have a similar delay dependency. We can actually improve the delay bounds of SRR from $O(N)\frac{1}{R_i}$ to $O(g \log N)\frac{1}{R_i}$ since E_i^{GSRR} is $O(g \log N)$. $O(g)$ factor is the effect of the hierarchy: the WFI, and therefore the delay bounds in GRR, takes into account how far ahead of its fair share a group has gotten, whereas β_i only depends on how far behind a flow is in terms of its ideal service. Intuitively, while in a standalone server, packets arriving at an empty queue are delayed because packets from the same session received more service in a previous time period, with the grouping strategy, a flow within a group may observe increased delay because of extra service that was received by other flows in that group. WF^{2}Q is the only algorithm whose service error is bounded both above and below by a maximum size packet, and thus has an $O(1)$ WFI.

Assuming g is small, we can provide good delay bounds for the standalone GRR server with any intergroup scheduler, and good network end-to-end delay bounds when the router nodes employ GRR servers. It is shown in [7] that the end-to-end delay of a packet p_i^k through a network of K servers Ψ^n is bounded by $GRC^1(p_i^k) - a(p_i^k) + \frac{(K-1)L_{i,max}}{\phi_i} + \sum_{n=1}^{K}(\beta_i^n + \tau^{n,n+1})$, where $\tau^{n,n+1}$ is the propagation delay from Ψ^n to Ψ^{n+1}.

VII. Experimental Results

To show the performance of GRR in practice, we present some results that quantify the resulting service error for various combinations of weights and flows for some of the algorithms presented in Section V. We compare the service errors of WFQ, WF^{2}Q, and SRR against their respective GRR counterparts, GWFQ, GWF^{2}Q, and GSRR. For this purpose, we developed a scheduling simulator to examine the scheduling behavior of these different algorithms across hundreds of thousands of different combinations of flows with different weight values.

The simulator takes four inputs, the scheduling algorithm, the number of flows N, the total number of weights S, and the number of flow-weight combinations. The simulator randomly assigns weights to flows and scales the weight values to ensure that they add up to S. It then schedules the flows using the specified algorithm and tracks the resulting service error. The simulator runs the scheduler until the resulting schedule repeats, then computes the maximum (most positive) and minimum (most negative) service error across the nonrepeating portion of the schedule for the given set of flows and weight assignments. This process of random weight allocation and scheduler simulation is repeated for the specified number

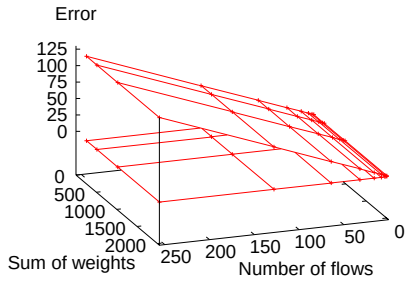

Fig. 1. WFQ service error

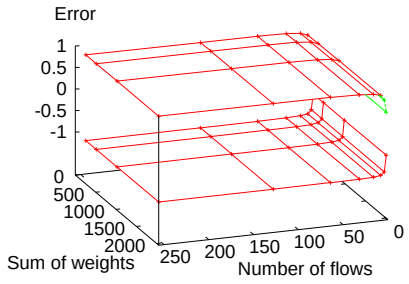

Fig. 2. WF^{2}Q service error

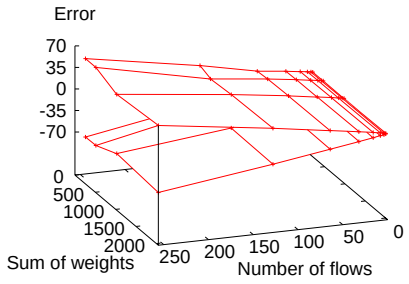

Fig. 3. SRR service error

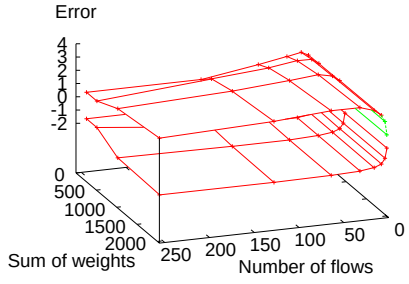

Fig. 4. GWFQ service error

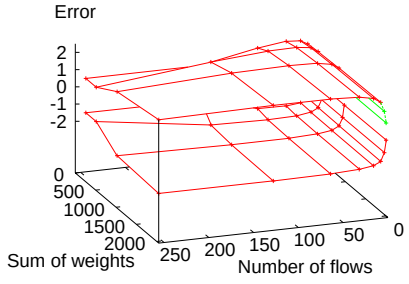

Fig. 5. GWF^{2}Q service error

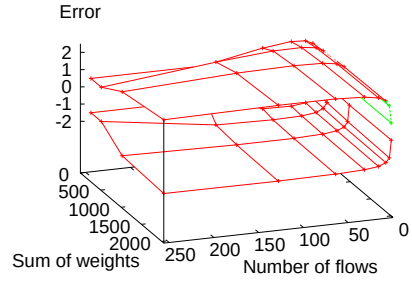

Fig. 6. GSRR service error

of flow-weight combinations. We then compute an average maximum service error and average minimum service error for the specified number of flow-weight combinations to obtain an "average-case" error range.

Since the absolute service error of a scheduler is often most clearly illustrated with skewed weight distributions, we ran simulations for each scheduling algorithm considered on 32 different combinations of N and S, with one of the flows given a weight equal to 50 percent of S. All of the other flows were then randomly assigned weights to sum to the remaining 50 percent of S. For each set of (N, S), we ran 2500 flow-weight combinations and determined the resulting average error ranges. For simplicity, all simulations were run with all flows backlogged at all times and all packets of equal size. The average service error ranges normalized by packet size for WFQ, WF^{2}Q, SRR, GWFQ, GWF^{2}Q, and GSRR with these skewed weight distributions are shown in Figures 1 to 6. Each figure shows two surfaces representing the maximum and minimum service error as a function of N and S for the respective scheduling algorithm.

Figure 1 shows the service error range for WFQ to be large, ranging between -1 to 128. WFQ has a lower bound of -1 but no constant upper bound on the error. In comparison, Figure 4 shows that GWFQ has significantly less service error than WFQ, ranging only from -2.13 to 2.79 while also preserving a constant lower bound. Figure 2 shows the service error range for WF^{2}Q, which is bounded between -1 to 1. In comparison, Figure 5 shows that GWF^{2}Q has only slightly larger service error ranging between -2.14 to 2.25, which is within the derived GWF^{2}Q constant service error bounds and is achieved with lower time complexity. Figure 3 shows the service error for SRR to be quite large, ranging between -96 to 96. In comparison, Figure 6 shows that GSRR has much less service error than SRR, ranging only between -2.14 to 2.26. These results quantitatively demonstrate the benefits GRR can provide in improving service error with often lower time complexity.

VIII. Conclusions

We have presented the design and analysis of Group Round-Robin, a packet scheduler that combines a novel grouping strategy with a specialized deficit round-robin algorithm to improve fairness and reduce time complexity. GRR is designed to utilize existing algorithms as the intergroup scheduler for scheduling among its groups. We proved that GRR only adds a constant factor to the relative fairness and absolute service error of any intergroup scheduler and also has low time complexity. As a result, we showed that GRR can be used to reduce the service error and time complexity of virtual time algorithms such as WFQ, SCFQ, SFQ, maintain the constant service error bound of WF^{2}Q while reducing its time complexity, and reduce the service error of SRR while maintaining its constant time complexity. Furthermore, we compare these approaches using consistent fairness measures and prove the fairness and service error bounds for several of

these algorithms. We implemented GRR and several GRR augmented algorithms and showed for various numbers of flows and weight distributions that GRR can reduce the service error of existing algorithms such as WFQ and SRR by well more than an order of magnitude. GRR's ability to narrow the tradeoff between fairness and computational complexity provides an effective packet scheduling mechanism for data networks.

APPENDIX

We prove four new results regarding the relative fairness and absolute service error bounds for various packet scheduling algorithms. These results are stated in the following four lemmas.

Lemma 3: For WFQ, the relative fairness $F_{i,j}$ between any two flows S_i and S_j is bounded by $\frac{L_{max}}{\phi_i} + \frac{L_{max}}{\phi_j}$.

Proof: We first make the important observation that for servers that reference GPS, such as WFQ, a flow is assumed backlogged for the purpose of fairness if it is backlogged under GPS. The distinction is important, since otherwise, we can easily construct an example where a flow backlogged only under WFQ receives half the service as another flow with the same weight. Consider any two flows S_i and S_j GPS-backlogged during the interval (t_1, t_2). If we denote by $f_n(t)$ the virtual finishing time of the packet at the head of $Q_{n,n\in\{i,j\}}$, then

$$f_i(t) - f_j(t) \leq \frac{L_{i,max}}{\phi_i} \tag{4}$$

Clearly, this difference is maximal just after a packet p_i^k from S_i is served, and stays fixed until another packet from S_i or S_j is served. Before p_i^k departed, $f_i(d(p_i^k) - 0) < f_j(d(p_i^k) - 0)$. Also, $f_i(d(p_i^k) + 0) = f_i(d(p_i^k) - 0) + \frac{L(p_i^{k+1})}{\phi_i}$ and $f_j(d(p_i^k) - 0) = f_j(d(p_i^k) + 0)$. It then follows that $f_i(d(p_i^k) + 0) < f_j(d(p_i^k) + 0) + \frac{L(p_i^{k+1})}{\phi_i}$, which proves (4). We observe that $f_n(t_1) = f_v(p_n^{k_1^n})$ and $f_n(t_2) = f_v(p_n^{k_2^n})$ where by $p_n^{k_m^n}$ we denote the last packet to have departed from flow S_n before time t_m.

Because for any packet arrival in the backlog interval, the virtual finishing time is $f_v(p_n^k) = f_v(p_n^{k-1}) + \frac{L(p_n^k)}{\phi_n}$, $n \in \{i, j\}$, we have $f_v(p_i^{k_2^i}) = f_v(p_i^{k_1^i}) + w_i(t_1, t_2)$ and $f_v(p_i^{k_2^j}) = f_v(p_i^{k_1^j}) + w_j(t_1, t_2)$. Subtracting, we get $w_i(t_1, t_2) - w_j(t_1, t_2) = f_v(p_i^{k_2^i}) - f_v(p_i^{k_1^i}) - (f_v(p_j^{k_2^j}) - f_v(p_j^{k_1^j})) = f_i(t_2) - f_j(t_2) + f_j(t_1) - f_i(t_1) \leq \frac{L_{i,max}}{\phi_i} + \frac{L_{i,max}}{\phi_j}$. Similarly, $w_j(t_1, t_2) - w_i(t_1, t_2) \leq \frac{L_{i,max}}{\phi_i} + \frac{L_{i,max}}{\phi_j}$. It follows that $F_{i,j}(t_1, t_2) = |w_i(t_1, t_2) - w_j(t_1, t_2)| \leq \frac{L_{i,max}}{\phi_i} + \frac{L_{i,max}}{\phi_j}$. ■

Lemma 4: For HS, the absolute service error E_i of flow S_i is bounded as follows: $E_i \leq (k-1)L_{max}$, where $k = \lceil \log(N) \rceil + 1$.

Proof: Let k be the height of the HS tree. Define $p^0 = S_i$, which is a leaf node, and p^j = parent(p^{j-1}) for $j = 1, ..., k-1$. For some $j \leq k-2$, $w_{p^j} - w_{p^{j+1}} = \frac{\phi_q(w_{p^j} - w_q)}{(\phi_{p^j} + \phi_q)}$ where q is the sibling of p^j. Since p^{j+1} always selects the child with the smaller normalized work, at any time, $w_{p^j} - w_q \leq \frac{L_{max}}{\phi_{p^j}}$ and $w_q - w_{p^j} \leq \frac{L_{max}}{\phi_q}$. Using $\phi_q = \phi_{p^{j+1}} - \phi_{p^j}$, we get $\frac{-L_{max}}{\phi_{p^{j+1}}} \leq w_{p^j} - w_{p^{j+1}} \leq \frac{L_{max}}{\phi_{p^j}} - \frac{L_{max}}{\phi_{p^{j+1}}}$. Since $w_i = w_{p^1}$ and $w = w_{p^{k-1}}$, we have $w_i - w = \sum_{j=0}^{k-2}(w_{p^j} - w_{p^{j+1}})$ which is then bounded above by $\sum_{j=0}^{k-2}(\frac{L_{max}}{\phi_{p^j}} - \frac{L_{max}}{\phi_{p^{j+1}}}) = L_{max}(\frac{1}{\phi_i} - \frac{1}{\phi})$. Also, $w_i = w_{p^1} \geq \sum_{j=0}^{k-2}(-\frac{L_{max}}{\phi_{p^{j+1}}}) \geq -L_{max}\sum_{j=0}^{k-3}\frac{1}{\phi_{p^1}} + \frac{L_{max}}{\phi} \geq -\frac{(k-1)L_{max}}{\phi_i}$, since $\phi_{p^i} \geq \phi_{p^1}$. Thus, the error $\phi_i(w_i - w)$ can get only as large as L_{max}, and cannot be less than $-(k-1)L_{max}$. Thus, $E_i = O(k) = O(\log N)$.

We will now show that this bound is tight. Consider the case with a flow S_1 of weight N and N flows of weight 1, and assume fixed packet sizes L for simplicity. The first time, S_1 is selected, and the work of the leaf p^0 corresponding to S_1 as well as the work of p^j, $j = 1, ..., k-1$ will become L. The next $k-1$ packets will be served from the siblings of p^j, $j = 0, ..., k-2$ since the $w_{p^j} > 0 = w_q$ where q is the sibling of p^j. Thus, after the first k packets have been served, the error of S_1 will be $|1 - \frac{k\phi_1}{\phi}| = |1 - \frac{kN}{2N}| = |1 - \frac{k}{2}|$. Thus, the error is worst case $\Omega(k) = \Omega(\log N)$. ■

Lemma 5: For HS, the relative fairness $F_{i,j}$ is bounded by $\frac{(k-1)L_{max}}{\min(\phi_i, \phi_j)} \leq \frac{\lceil \log(N) \rceil L_{max}}{\min(\phi_i, \phi_j)}$, where N is the number of flows being scheduled and $k = \lceil \log(N) \rceil + 1$.

Proof: Assuming without loss of generality that $w_i \geq w_j$, using the results in Lemma 4, we have $F_{i,j} = w_i - w_j = w_i - w + w - w_j \leq L_{max}(\frac{1}{\phi_i} - \frac{1}{\phi}) + \frac{(k-2)L_{max}}{\phi_j} + \frac{L_{max}}{\phi} \leq \frac{(k-1)L_{max}}{\min(\phi_i, \phi_j)}$. ■

Lemma 6: For SRR, the absolute service error E_i of flow S_i is bounded as follows: $E_i \leq \frac{(N+1)k}{2} = O(Nk)$, where N is the number of flows being scheduled and k is the number of bits needed to store the maximum weight of any flow.

Proof: We will first show a $O(Nk)$ bounds for E_i of SRR. For this, we introduce the equivalent complete binary tree associated with the WSS. This tree has k levels, and each node on level j corresponds to the WSS number j (where 'k' is the root and '1' are the leaves). We notice that the inorder traversal of this tree corresponds to the k^{th} order WSS (this observation will allow us to later

describe an algorithm to dynamically generate the WSS). In the following discussion, we assume $L = L_{max} = 1$. The error bounds can be scaled afterwards by L_{max}.

Consider flow S_i with weight ϕ_i. Since the leaves in the WSS-tree are visited every other time, we can assume that the node currently visited is a leaf node (otherwise, the error can increase by at most 1). Let β be the number of leaves considered thus far. Then the number of level j nodes visited is $\lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor$, as can be deduced from the WSS-tree. Then the work done for flow S_i is $W_i = \sum_{j=1}^{k} N_j^i \lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor$ while the total work done is $W = \sum_{j=1}^{k} N_j \lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor$ where N_j is the number of flows that have a non-zero bit at position j, and N_j^i is the j^{th} bit of ϕ_i. Thus, $W_i - (\frac{\phi_i}{\phi})W = \sum_{j=1}^{k} N_j^i \lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor - \frac{\sum_j N_j^i 2^{k-j}}{\sum_j N_j 2^{k-j}} \sum_j N_j \lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor$. Then $W_i - (\frac{\phi_i}{\phi})W \leq \sum_j N_j^i (\frac{\beta}{2^{j-1}} - \frac{1}{2}) - \frac{\sum_j N_j^i 2^{-j}}{\sum_j N_j 2^{-j}} \sum_j N_j (\frac{\beta}{2^{j-1}} + \frac{1}{2}) = \frac{\sum_j N_j^i}{2} + \frac{\sum_j N_j^i 2^{-j}}{\sum_j N_j 2^{-j}} \frac{\sum_j N_j}{2} \leq \frac{k}{2} + \frac{Nk}{2}$, since $\sum_j N_j^i \leq k$ and $\sum_j N_j \leq Nk$ (each of ϕ_i can have at most k bits on). Also, $W_i - (\frac{\phi_i}{\phi})W \geq \sum_j N_j^i (\frac{\beta}{2^{j-1}} + \frac{1}{2}) - \frac{\sum_j N_j^i 2^{-j}}{\sum_j N_j 2^{-j}} \sum_j N_j (\frac{\beta}{2^{j-1}} - \frac{1}{2}) = -\frac{\sum_j N_j^i}{2} - \frac{\sum_j N_j^i 2^{-j}}{\sum_j N_j 2^{-j}} \frac{\sum_j N_j}{2} \geq \frac{-k}{2} - \frac{Nk}{2}$. Therefore, $E_i = |W_i - (\frac{\phi_i}{\phi})W| \leq \frac{k(N+1)}{2} = O(Nk)$.

We will now show that in fact E_i can be $\Omega(kN)$. For this, consider the example with $N + 1$ flows, S_1 through S_N, where ϕ_1 = 11...1 101...010$_2$ = $2^k - \frac{2^{\frac{k}{2}}+2}{3}$ and $\phi_{i,i>1}$ = 00...0 010...101$_2$ = $\frac{2^{\frac{k}{2}}-1}{3}$ (we assume k is a multiple of 4). The bit representations of ϕ_i satisfy the following: the first $\frac{k}{2}$ bits are 1 for ϕ_1 and 0 for $\phi_{i,i>1}$, while bit $j > \frac{k}{2}$ is $j \bmod 2$ for ϕ_1 and $(j+1) \bmod 2$ for $\phi_{i,i>1}$, where bit 1 is the most significant bit. Let p be the level $\frac{k}{2}$ node in the WSS-tree such that in the path from the root to p, odd level nodes are right children, and even level nodes are left children. For each node, because of the inorder traversal of the WSS-tree, the left subtree will have been visited before the node, and the right subtree is visited after the node. Thus, at the moment we are considering, only the even level nodes in the path and their left subtrees will have been visited. The following relation holds for the work done during a completely visited subtree rooted at a node on level $\frac{k}{2} + j$, j odd: $W_1\phi - W\phi_1 < 2^k \frac{N}{3} - 2^{\frac{k}{2}+j} \frac{N}{3}$. ($W_1\phi - W\phi_1 = W_1(\phi - \phi_1) - (W - W_1)\phi_1 = ((2^{\frac{k}{2}+j}) - \frac{(2^j+1)}{3})((2^{\frac{k}{2}-1})\frac{N}{3}) - ((2^j-1)\frac{N}{3})(2^k - \frac{(2^{\frac{k}{2}}+2)}{3}) = 2^{k+j}\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{9} - 2^j\frac{N}{9} - 2^{\frac{k}{2}}\frac{N}{9} + \frac{N}{9} - 2^{k+j}\frac{N}{3} + 2^k\frac{N}{3} + 2^{\frac{k}{2}+j}\frac{N}{9} - 2^{\frac{k}{2}}\frac{N}{9} - \frac{2N}{9} = 2^k\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{3} + 2^j\frac{N}{3} - 2^{\frac{k}{2}+1}\frac{N}{9} - \frac{N}{9} \leq 2^k\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{3}$).

Then for all the completely visited subtrees up to the time node p is considered, the difference $W_1\phi - W\phi_1$ is less than $\sum_{j=1}^{\frac{k}{2}-1}(2^k\frac{N}{3}) - \sum_{j=1}^{\frac{k}{2}-1}(2^{\frac{k}{2}+j}\frac{N}{3}) = k2^{k-1}\frac{N}{3} - (2^{k+1} - 2^{\frac{k}{2}+1})\frac{N}{3}$. The S_1 error at node p is then $E_1 = W_1 - (\frac{\phi_1}{\phi})(W + N\frac{k}{2}) = \frac{W_1\phi - W\phi_1 - \phi_1 N\frac{k}{2}}{\phi} \leq (\frac{1}{\phi})(k2^{k-1}\frac{N}{3} - (2^{k+1} - 2^{\frac{k}{2}+1})\frac{N}{3} - (2^k - \frac{2^{\frac{k}{2}}+2}{3})k\frac{N}{2}) \leq (\frac{1}{\phi})(N2^{k-1})(\frac{k}{3} - k) \leq -(\frac{1}{\phi})(\frac{k2^kN}{3}) \leq -\frac{(k2^k\frac{N}{3})}{2^k - \frac{(2^{\frac{k}{2}}+2)}{3} + \frac{N(2^{\frac{k}{2}}-1)}{3}} \leq -\frac{Nk}{6}$ provided that $N \leq 2^{k/2}$. Clearly, if $N > 2^{k/2}$, then we can use the N, 1, 1, ..., 1 (N times) example to show that the error is $\Omega(N)$, which is unacceptably large by itself. ■

O(1) WSS dynamic generation: A way to generate the WSS is by means of traversing the WSS-tree described in Lemma 6. We describe an easy algorithm to accomplish this: keep a boolean array $A[1, ..., k]$, initialized to false. At each step, select level 1, and then find the least $j > 1$ such that $A[j]$ is false. Reset all $A[i]$ with $i < j$ to false, select level j, and set $A[j]$ to true. This method of generating the WSS is worst case k for a step, but on average, it is 2. To see that, note that the total work is $2^{k-1} + 2^{k-2}2 + 2^{k-3}3 + ... + 2^0 k \leq 2^k(\sum_{n=0}^{\infty} \frac{n}{2^n}) = 2^{k+1}$. Since $2^{k-1} + .. + 1 = 2^k - 1$, the average work for each number in the WSS is $\frac{2^{k+1}}{2^k-1} \approx 2$. Since we know that the sequence in the WSS doesn't change, we can amortize the cost of computing the WSS by performing 2 operations instead of 1 each step and filling up a buffer from which we select the next number of the WSS. The buffer will need to be only of size k, and not $3 * 2^{k/2}$ as [3] requires for its static WSS. We can show that the buffer always contains between 1 and k WSS numbers, so that no more than 2 operations need to be performed each step.

REFERENCES

[1] J. Bennett and H. Zhang, "WF^{2}Q: Worst-case Fair Weighted Fair Queueing," in *Proceedings of INFOCOM '96*, San Francisco, CA, Mar. 1996.

[2] J. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms ," in *Proceedings of ACM SIGCOMM '96*, Aug. 1996

[3] G. Chuanxiong, "SRR: An 0(1) Time Complexity Packet Scheduler for Flows in Multi-Service Packet Networks," in *Proceedings of ACM SIGCOMM '01*, Aug. 2001.

[4] J. Cobb, M. Gouda, and A. El-Nahas, "Time-Shift Scheduling - Fair Scheduling of Flows in High-Speed Networks," in *IEEE/ACM Transactions on Networking*, June 1998.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in *Proceedings of ACM SIGCOMM '89*, Austin, TX, Sept. 1989.

[6] S. J. Golestani, " A Self-Clocked Fair Queueing Scheme for Broadband applications," in *Proceedings of IEEE INFOCOM '94*, Apr. 1994.

[7] P. Goyal, S. Lam, and H. Vin, "Determining End-to-End Delay Bounds in Heterogeneous Networks," in *Proceedings NOSSDAV*, Apr. 1995.

[8] P. Goyal, H. Vin, and H. Cheng, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," in *IEEE/ACM Transactions on Networking*, Oct. 1997.

[9] L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*. New York: John Wiley & Sons, 1976.

[10] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, 1(3), June 1993.

[11] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round-Robin," in *Proceedings of ACM SIGCOMM '95*, 4(3), Sept. 1995.

[12] D. Stiliadis, and A. Varma, "Efficient Fair Queueing Algorithms for Packet-Switched Networks," in *IEEE/ACM Transactions on Networking*, Apr. 1998.

[13] C. Waldspurger, *Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Sept. 1995.

[14] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switched Networks," in *ACM Transactions on Computer Systems*, 9(2), May 1991.